User Interface - Bug #12112

Memory leak in client JVM when deleting dynamic frames with dynamic widgets

11/03/2026 08:14 AM - Şerban Bursuc

Status:	WIP	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Şerban Bursuc	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

Related issues:

Related to User Interface - Bug #10530: Web client cursor gets stuck on the r...	**WIP**

History

#2 - 11/03/2026 08:15 AM - Şerban Bursuc

- Related to Bug #10530: Web client cursor gets stuck on the resize cursor unexpectedly added

#3 - 11/03/2026 08:23 AM - Şerban Bursuc

This should affect all the customer projects that use UI and also create dynamic windows. It is not a regression and has been present for a while at the very least. Found during #10530 regression testing using cursor related testcases. Report below.

Symptom

A window created at runtime and then deleted (CREATE WINDOW, DELETE WIDGET) stays in the memory of the FWD client process, together with its frames, the widgets in them and their mouse actions. Nothing frees them later: every deleted window adds to the client heap, so a session that keeps opening and closing windows grows for as long as it runs. The windows do not have to be hovered or clicked, and no mouse pointer is involved; creating and deleting them is enough.

Per deleted window with one frame and three buttons, these stay live in the client JVM:

Class	Instances per deleted window
WindowGuiImpl	1
FrameGuiImpl	1
ButtonGuiImpl	3
MouseHoverAction	3
MouseEmptySelector	1

Found while working on #10530. Same pattern as #10322.

Testcase

Each choose of the Cycle button creates and deletes 25 windows, each with a frame and three buttons:

```
/* Each choose of Cycle creates and deletes 25 windows, each with a frame and three buttons. */
DEFINE VARIABLE hWin AS HANDLE  NO-UNDO.
DEFINE VARIABLE hFrm AS HANDLE  NO-UNDO.
DEFINE VARIABLE hBtn AS HANDLE  NO-UNDO.
DEFINE VARIABLE i    AS INTEGER NO-UNDO.
DEFINE VARIABLE j    AS INTEGER NO-UNDO.
```

```
DEFINE BUTTON bCycle LABEL "Cycle".
DEFINE FRAME f bCycle.

ON CHOOSE OF bCycle
DO:
   DO i = 1 TO 25:
      CREATE WINDOW hWin
         ASSIGN WIDTH   = 40
                HEIGHT  = 10
                VISIBLE = TRUE.
      CREATE FRAME hFrm
         ASSIGN PARENT  = hWin
                WIDTH   = 38
                HEIGHT  = 8
                VISIBLE = TRUE.
      DO j = 1 TO 3:
         CREATE BUTTON hBtn
            ASSIGN FRAME     = hFrm
                   ROW       = j * 2
                   COLUMN    = 3
                   LABEL     = "B" + STRING(j)
                   SENSITIVE = TRUE
                   VISIBLE   = TRUE.
      END.
      DELETE WIDGET hWin.
   END.
END.

ENABLE bCycle WITH FRAME f.
WAIT-FOR WINDOW-CLOSE OF CURRENT-WINDOW.
```

Steps, FWD trunk rev 16818, web GUI client:

1. Run the testcase and log in.
2. Take the histogram below.
3. Choose Cycle (25 windows created and deleted), take the histogram again.
4. Choose Cycle again (50 windows), take the histogram again.

Measurement

Run jmap as the OS user the client JVM runs under. jmap -histo:live runs a full garbage collection first, so it counts only objects that are still reachable:

```
PID=$(pgrep -n -f '^java .*ClientDriver')
jmap -histo:live $PID | grep -E '#instances|^---|[.](WindowGuiImpl|FrameGuiImpl|ButtonGuiImpl|MouseHoverAction
|MouseEmptySelector)$'
```

After the login (the default window, its frame and the Cycle button):

```
 num     #instances         #bytes  class name (module)
-------------------------------------------------------
 629:             1             448  com.goldencode.p2j.ui.client.gui.FrameGuiImpl
 800:             1             264  com.goldencode.p2j.ui.client.gui.WindowGuiImpl
1011:             1             160  com.goldencode.p2j.ui.client.gui.ButtonGuiImpl
2011:             1              40  com.goldencode.p2j.ui.client.gui.driver.MouseEmptySelector
2013:             1              40  com.goldencode.p2j.ui.client.gui.driver.MouseHoverAction
```

After 25 windows created and deleted:

```
 num     #instances         #bytes  class name (module)
-------------------------------------------------------
 126:            76          12160  com.goldencode.p2j.ui.client.gui.ButtonGuiImpl
 131:            26          11648  com.goldencode.p2j.ui.client.gui.FrameGuiImpl
 174:            26           6864  com.goldencode.p2j.ui.client.gui.WindowGuiImpl
 265:            76           3040  com.goldencode.p2j.ui.client.gui.driver.MouseHoverAction
```

```
 452:            26           1040  com.goldencode.p2j.ui.client.gui.driver.MouseEmptySelector
```

After 50 windows created and deleted:

```
 num     #instances         #bytes  class name (module)
-------------------------------------------------------
  93:           151          24160  com.goldencode.p2j.ui.client.gui.ButtonGuiImpl
  97:            51          22848  com.goldencode.p2j.ui.client.gui.FrameGuiImpl
 127:            51          13464  com.goldencode.p2j.ui.client.gui.WindowGuiImpl
 203:           151           6040  com.goldencode.p2j.ui.client.gui.driver.MouseHoverAction
 331:            51           2040  com.goldencode.p2j.ui.client.gui.driver.MouseEmptySelector
```

Every deleted window remains: each 25 windows add 25 windows, frames and selectors and 75 buttons and hover actions. The histograms only list a few classes; everything these widgets reference stays as well.

Cause

Deleting the window calls destroy() for its own parts (title bar, caption buttons, status line, message area) but never for its frames and their widgets, so MouseHandler.deregisterWidget never runs for them. Their actions stay in the MouseHandler registry (WorkArea.actions): the buttons' MouseHoverAction (from AbstractWidget._setEnabled) and the frame's MouseEmptySelector (from AbstractGuiDriver.registerEmptySelector). Each action holds its widget (MouseWidgetAction.widget), which keeps the whole window reachable.

#4 - 11/03/2026 08:31 AM - Șerban Bursuc

- Assignee set to Șerban Bursuc

If this hasn't been fixed yet in a branch I can take a look at this.

#5 - 11/03/2026 09:03 AM - Constantin Asofiei

Serban, please confirm the test in 4GL - what does valid-handle say for the hFrm and hBtn widgets? Because to me it doesn't look like is FWD's responsibility to clean this up.

#6 - 11/03/2026 09:22 AM - Șerban Bursuc

Constantin Asofiei wrote:

> Serban, please confirm the test in 4GL - what does valid-handle say for the hFrm and hBtn widgets? Because to me it doesn't look like is FWD's responsibility to clean this up.

Didn't consider the possibility of the 4GL being bad and you're right, 4GL does not delete the frame and its content either after deleting the window widget handle. This is an AI test generation issue, it should probably be put in a skill somewhere to do a proper cleanup.

There's still a difference though - when adding DELETE WIDGET hFrm to the testcase the client side memory is fixed but the server side seems to still grow. jmap shows 1 WindowWidget, 1 FrameWidget and 2 ButtonWidget widgets after each cycle. In OE all widgets become invalid when queried with VALID-HANDLE. I will draft the report once I double check the issue.

#7 - 11/03/2026 09:33 AM - Șerban Bursuc

The findings again look legit. I opened access to the remote-oe machine for this to check. If the data collected is right then this is a FWD bug.

Constantin, please take a look.

Symptom

DELETE WIDGET of a dynamic frame deletes only the first dynamic widget in it. Its other dynamic widgets stay valid (VALID-HANDLE returns yes) and stay in the memory of the FWD server, together with the deleted frame. OpenEdge deletes all the widgets of a deleted frame. A session which keeps creating and deleting dynamic frames grows the server heap for as long as it runs; the FWD client is not affected.

Per deleted frame with three dynamic buttons, these stay live in the server JVM:

Class	Instances per deleted frame
FrameWidget	1
FieldGroup	2
ButtonWidget	2
FrameConfig	1
ButtonConfig	4

Found while working on #10530. Deleting a window does not delete its frames, in OpenEdge either: they and their widgets stay valid, so the testcase deletes the frame itself.

Testcase

Each choose of Cycle creates 25 dynamic frames with three dynamic buttons each, deletes every frame and shows how many of the deleted buttons are still valid:

```
/* Each choose of Cycle creates 25 dynamic frames with three dynamic buttons each and deletes every frame.
   Deleting a frame deletes its widgets: no button may stay valid. */
DEFINE VARIABLE hFrm    AS HANDLE    NO-UNDO.
DEFINE VARIABLE hBtn    AS HANDLE    NO-UNDO EXTENT 3.
DEFINE VARIABLE i       AS INTEGER   NO-UNDO.
DEFINE VARIABLE j       AS INTEGER   NO-UNDO.
DEFINE VARIABLE nBtn    AS INTEGER   NO-UNDO.
DEFINE VARIABLE nValid  AS INTEGER   NO-UNDO.
DEFINE VARIABLE cStatus AS CHARACTER NO-UNDO FORMAT "x(30)" LABEL "Still valid".

PAUSE 0 BEFORE-HIDE.

DEFINE BUTTON bCycle LABEL "Cycle".
DEFINE FRAME f bCycle cStatus WITH SIDE-LABELS.

ON CHOOSE OF bCycle
DO:
   DO i = 1 TO 25:
      CREATE FRAME hFrm
         ASSIGN PARENT  = CURRENT-WINDOW
                ROW     = 4
                WIDTH   = 38
                HEIGHT  = 8
```

```
                VISIBLE = TRUE.
      DO j = 1 TO 3:
         CREATE BUTTON hBtn[j]
            ASSIGN FRAME     = hFrm
                   ROW       = j * 2
                   COLUMN    = 3
                   LABEL     = "B" + STRING(j)
                   SENSITIVE = TRUE
                   VISIBLE   = TRUE.
      END.
      DELETE WIDGET hFrm.
      DO j = 1 TO 3:
         nBtn = nBtn + 1.
         IF VALID-HANDLE(hBtn[j]) THEN
            nValid = nValid + 1.
      END.
   END.
   cStatus = STRING(nValid) + " of " + STRING(nBtn) + " buttons".
   DISPLAY cStatus WITH FRAME f.
END.

CURRENT-WINDOW:TITLE = "Frame delete".
ENABLE bCycle WITH FRAME f.
WAIT-FOR WINDOW-CLOSE OF CURRENT-WINDOW.
```

PAUSE 0 BEFORE-HIDE only keeps OpenEdge from pausing before it hides each deleted frame.

Run it and choose Cycle twice:

	OpenEdge 11.6 (prowin, 64-bit)	FWD (web GUI)
after one choose	Still valid: 0 of 75 buttons	Still valid: 50 of 75 buttons
after two	Still valid: 0 of 150 buttons	Still valid: 100 of 150 buttons

Only the first button of each frame is deleted. The same happens with more buttons (with four, B2 to B4 stay valid) and when the frame's window was deleted before the frame. FWD was branch 10530a rev 16820, whose changes are client side only; HandleOps.java is the same as in trunk rev 16818.

Measurement

jmap -histo:live runs a full garbage collection first, so it counts only objects that are still reachable. Run it as the OS user of the JVM; jcmd -l lists the JVMs (the server is FwdLauncher server.xml, the session's client ClientDriver):

```
jmap -histo:live <server pid> | grep -E '#instances|^---|[.](ButtonWidget|FrameWidget|FieldGroup|ButtonConfig|
FrameConfig)$'
jmap -histo:live <client pid> | grep -E '#instances|^---|[.](WindowGuiImpl|FrameGuiImpl|ButtonGuiImpl|MouseHov
erAction|MouseEmptySelector)$'
```

Server, after the login (the static frame f and the Cycle button):

```
 num     #instances         #bytes  class name (module)
-------------------------------------------------------
 570:             2             944  com.goldencode.p2j.ui.FrameConfig
 620:             2             768  com.goldencode.p2j.ui.ButtonConfig
 881:             2             320  com.goldencode.p2j.ui.FieldGroup
1412:             1             152  com.goldencode.p2j.ui.ButtonWidget
1433:             1             144  com.goldencode.p2j.ui.FrameWidget
```

Server, after 25 frames created and deleted:

```
 num     #instances         #bytes  class name (module)
-------------------------------------------------------
  92:           103           39552  com.goldencode.p2j.ui.ButtonConfig
 158:            27           12744  com.goldencode.p2j.ui.FrameConfig
```

```
 205:            52           8320  com.goldencode.p2j.ui.FieldGroup
 213:            52           7904  com.goldencode.p2j.ui.ButtonWidget
 297:            26           3744  com.goldencode.p2j.ui.FrameWidget
```

Server, after 50 frames created and deleted:

```
 num     #instances         #bytes  class name (module)
-------------------------------------------------------
  65:           203          77952  com.goldencode.p2j.ui.ButtonConfig
 121:            52          24544  com.goldencode.p2j.ui.FrameConfig
 146:           102          16320  com.goldencode.p2j.ui.FieldGroup
 150:           102          15504  com.goldencode.p2j.ui.ButtonWidget
 220:            51           7344  com.goldencode.p2j.ui.FrameWidget
```

Client, after 50 frames created and deleted (the same counts as after the login):

```
 num     #instances         #bytes  class name (module)
-------------------------------------------------------
 593:             1            448  com.goldencode.p2j.ui.client.gui.FrameGuiImpl
 758:             1            264  com.goldencode.p2j.ui.client.gui.WindowGuiImpl
 944:             1            160  com.goldencode.p2j.ui.client.gui.ButtonGuiImpl
1338:             2             80  com.goldencode.p2j.ui.client.gui.driver.MouseHoverAction
1873:             1             40  com.goldencode.p2j.ui.client.gui.driver.MouseEmptySelector
```

Cause

HandleOps.delete(handle, boolean) (added for #8000) deletes the dynamic children of a deleted container by walking firstChild() and getNextSibling(), but it reads the next sibling only when the child is a CommonHandleTree (frame, field group, window, menu). A button is not one, so the walk stops after the first widget of the frame's field group. The widgets left valid keep the deleted frame and its field groups reachable.

#8 - 11/03/2026 09:42 AM - Constantin Asofiei

Understood, go ahead and fix this.

#9 - 11/03/2026 09:44 AM - Șerban Bursuc

- Subject changed from Memory leak in client JVM caused by deleted dynamic windows persisting to Memory leak in client JVM when deleting dynamic frames with dynamic widgets

- Status changed from New to WIP